PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ● The ACM Digital Library ○ The Guide

garbage collection

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used **garbage collection** Found **28,651** of **139,567**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown

Relevance scale ☐ ▭ ▤ ▦ ■

1 Systems: Myths and realities: the performance impact of garbage collection
Stephen M. Blackburn, Perry Cheng, Kathryn S. McKinley
June 2004 **Proceedings of the joint international conference on Measurement and modeling of computer systems**
Full text available: pdf(305.06 KB) Additional Information: full citation, abstract, references, index terms

This paper explores and quantifies garbage collection behavior for three whole heap collectors and generational counterparts: *copying semi-space, mark-sweep,* and *reference counting*, the canonical algorithms from which essentially all other collection algorithms are derived. Efficient implementations in MMTk, a Java memory management toolkit, in IBM's Jikes RVM share all common mechanisms to provide a clean experimental platform. Instrumentation separates collector and program behav ...

Keywords: generational, java, mark-sweep, reference counting, semi-space

2 Comparing mark-and sweep and stop-and-copy garbage collection
Benjamin Zorn
May 1990 **Proceedings of the 1990 ACM conference on LISP and functional programming**
Full text available: pdf(1.02 MB) Additional Information: full citation, abstract, references, citings, index terms

Stop-and-copy garbage collection has been preferred to mark-and-sweep collection in the last decade because its collection time is proportional to the size of reachable data and not to the memory size. This paper compares the CPU overhead and the memory requirements of the two collection algorithms extended with generations, and finds that mark-and-sweep collection requires at most a small amount of additional CPU overhead (3-6%) but, requires an average of 20% (and up to 40%) less memory t ...

3 Connectivity-based garbage collection
Martin Hirzel, Amer Diwan, Matthew Hertz
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications**, Volume 38 Issue 11
Full text available: pdf(521.65 KB) Additional Information: full citation, abstract, references, index terms

We introduce a new family of connectivity-based garbage collectors (Cbgc) that are based on potential object-connectivity properties. The key feature of these collectors is that the

PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ◉ The ACM Digital Library ○ The Guide

garbage collection and descriptors and stack and heap and poi

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used
garbage collection and **descriptors** and **stack** and **heap** and **pointers**

Found **13,107** of **139,567**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Combining region inference and garbage collection
Niels Hallenberg, Martin Elsman, Mads Tofte
May 2002 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation**, Volume 37 Issue 5
Full text available: pdf(195.49 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes a memory discipline that combines region-based memory management and copying garbage collection by extending Cheney's copying garbage collection algorithm to work with regions. The paper presents empirical evidence that region inference very significantly reduces the number of garbage collections; and evidence that the fastest execution is obtained by using regions alone, without garbage collection. The memory discipline is implemented for Standard ML in the ML Kit compiler ...

Keywords: garbage collection, region interface, standard ML

2 On the usefulness of type and liveness accuracy for garbage collection and leak detection
Martin Hirzel, Amer Diwan, Johannes Henkel
November 2002 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 24 Issue 6
Full text available: pdf(684.85 KB) Additional Information: full citation, abstract, references, index terms

The effectiveness of garbage collectors and leak detectors in identifying dead objects depends on the *accuracy* of their reachability traversal. Accuracy has two orthogonal dimensions: (i) whether the reachability traversal can distinguish between pointers and nonpointers (*type accuracy*), and (ii) whether the reachability traversal can identify memory locations that will be dereferenced in the future (*liveness accuracy*). This article presents an experimental study of the impo ...

Keywords: Conservative garbage collection, leak detection, liveness accuracy, program analysis, type accuracy

3 Compact garbage collection tables
David Tarditi
October 2000 **ACM SIGPLAN Notices , Proceedings of the second international**

PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ● The ACM Digital Library ○ The Guide

garbage collection and descriptors and stack and heap and poi

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used
garbage collection and **descriptors** and **stack** and **heap** and **pointers** and **call sites**

Found **27,994** of **139,567**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown

Relevance scale

1 On the usefulness of type and liveness accuracy for garbage collection and leak detection
Martin Hirzel, Amer Diwan, Johannes Henkel
November 2002 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 24 Issue 6
Full text available: pdf(684.85 KB) Additional Information: full citation, abstract, references, index terms

The effectiveness of garbage collectors and leak detectors in identifying dead objects depends on the *accuracy* of their reachability traversal. Accuracy has two orthogonal dimensions: (i) whether the reachability traversal can distinguish between pointers and nonpointers (*type accuracy*), and (ii) whether the reachability traversal can identify memory locations that will be dereferenced in the future (*liveness accuracy*). This article presents an experimental study of the impo ...

Keywords: Conservative garbage collection, leak detection, liveness accuracy, program analysis, type accuracy

2 Compact garbage collection tables
David Tarditi
October 2000 **ACM SIGPLAN Notices , Proceedings of the second international symposium on Memory management**, Volume 36 Issue 1
Full text available: pdf(958.92 KB) Additional Information: full citation, abstract, index terms

Garbage collection tables for finding pointers on the stack can be represented in 20-25% of the space previously reported. Live pointer information is often the same at many call sites because there are few pointers live across most call sites. This allows live pointer information to be represented compactly by a small index into a table of descriptions of pointer locations. The mapping from program counter values to those small indexes can be represented compactly using several techniques. T ...

3 Garbage collection for strongly-typed languages using run-time type reconstruction
Shail Aditya, Christine H. Flood, James E. Hicks
July 1994 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1994 ACM conference on LISP and functional programming**, Volume VII Issue 3
Full text available: pdf(1.40 MB) Additional Information: full citation, abstract, references, citings, index terms, review

PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ● The ACM Digital Library ○ The Guide

garbage collection and descriptors and stack and heap and poi

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction s

Terms used
garbage collection and **descriptors** and **stack** and **heap** and **pointers** and **call sites** and **reference table regis**

Sort results by relevance
Display results expanded form

Save results to a Binder
Search Tips
☐ Open results in a new window

Try an Advanced Search
Try this search in The ACM Gui

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown
Relevance scal

1 Compact garbage collection tables
David Tarditi
October 2000 **ACM SIGPLAN Notices , Proceedings of the second international symposium on Memory management**, Volume 36 Issue 1
Full text available: pdf(958.92 KB) Additional Information: full citation, abstract, index terms

Garbage collection tables for finding pointers on the stack can be represented in 20-25% of the sp previously reported. Live pointer information is often the same at many call sites because there ar pointers live across most call sites. This allows live pointer information to be represented compactl small index into a table of descriptions of pointer locations. The mapping from program counter va those small indexes can be represented compactly using several techniques. T ...

2 On the usefulness of type and liveness accuracy for garbage collection and leak detection
Martin Hirzel, Amer Diwan, Johannes Henkel
November 2002 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 6
Full text available: pdf(684.85 KB) Additional Information: full citation, abstract, references, index terms

The effectiveness of garbage collectors and leak detectors in identifying dead objects depends on t *accuracy* of their reachability traversal. Accuracy has two orthogonal dimensions: (i) whether the reachability traversal can distinguish between pointers and nonpointers (*type accuracy*), and (ii) w the reachability traversal can identify memory locations that will be dereferenced in the future (*liv accuracy*). This article presents an experimental study of the impo ...

Keywords: Conservative garbage collection, leak detection, liveness accuracy, program analysis, accuracy

3 Fine-grained mobility in the Emerald system
Eric Jul, Henry Levy, Norman Hutchinson, Andrew Black
February 1988 **ACM Transactions on Computer Systems (TOCS)**, Volume 6 Issue 1
Full text available: pdf(2.01 MB) Additional Information: full citation, abstract, references, citings, index terms,

Emerald is an object-based language and system designed for the construction of distributed prog An explicit goal of Emerald is support for object mobility; objects in Emerald can freely move withi system to take advantage of distribution and dynamically changing environments. We say that Em has fine-grained mobility because Emerald objects can be small data objects as well as process ob

Fine-grained mobility allows us to apply mobility in new ways but presents implemen ...

4 Efficient and safe-for-space closure conversion
Zhong Shao, Andrew W. Appel
January 2000 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 2:
Full text available: pdf(336.90 KB) Additional Information: full citation, abstract, references, citings, index terms

Modern compilers often implement function calls (or returns) in two steps: first, a "closure" enviro
is properly installed to provide access for free variables in the target program fragment; second, tl
control is transferred to the target by a "jump with arguments (for results)." Closure conversion—'
decides where and how to represent closures at runtime—is a crucial step in the compilation of fur
languages. This paper presents a new alg ...

Keywords: callee-save registers, closure conversion, closure representation, compiler optimizatio
analysis, heap-based compilation, space safety

5 A typed interface for garbage collection
Joseph C. Vanderwaart, Karl Crary
January 2003 **ACM SIGPLAN Notices , Proceedings of the 2003 ACM SIGPLAN international workshop on Types in languages design and implementation**, Volume 38 Issue 3
Full text available: pdf(316.77 KB) Additional Information: full citation, abstract, references, index terms

An important consideration for certified code systems is the interaction of the untrusted program v
runtime system, most notably the garbage collector. Most certified code systems that treat the ga
collector as part of the trusted computing base dispense with this issue by using a collector whose
interface with the program is simple enough that it does not pose any certification challenges. Hov
this approach rules out the use of many sophisticated high-performance garbage collec ...

Keywords: certified code, garbage collection, type systems, typed compilation

6 Tag-free garbage collection using explicit type parameters
Andrew Tolmach
July 1994 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1994 ACM conference on LISP : functional programming**, Volume VII Issue 3
Full text available: pdf(1.04 MB) Additional Information: full citation, abstract, references, citings, index terms

We have constructed a practical tag-free garbage collector based on explicit type parameterizatior
polymorphic functions, for a dialect of ML. The collector relies on type information derived from an
explicitly-typed 2nd-order representation of the program, generated by the compiler as a byprodu
ordinary Hindley-Milner type inference. Runtime type manipulations are performed lazily to minimi
execution overhead. We present details of our implementation approach, and preliminary per ...

7 Garbage collection for a client-server persistent object store
Laurent Amsaleg, Michael J. Franklin, Olivier Gruber
August 1999 **ACM Transactions on Computer Systems (TOCS)**, Volume 17 Issue 3
Full text available: pdf(267.18 KB) Additional Information: full citation, abstract, references, citings, index terms,

We describe an efficient server-based algorithm for garbage collecting persistent object stores in a
server environmnet. The algorithm is incremental and runs concurrently with client transactions. U
previous algorithms, it does not hold any transactional locks on data and does non require callbacl
clients. It is fault-tolerant, but performs very little logging. The algorithm has been designed to be
integrated into existing systems, and therefore it works with standard i ...

Keywords: client-server system, logging, persistent object-store, recovery

8 Parallel execution of prolog programs: a survey
Gopal Gupta, Enrico Pontelli, Khayri A.M. Ali, Mats Carlsson, Manuel V. Hermenegildo
July 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 23
Full text available: pdf(1.95 MB) Additional Information: full citation, abstract, references, citings, index terms

Since the early days of logic programming, researchers in the field realized the potential for exploi
of parallelism present in the execution of logic programs. Their high-level nature, the presence of
nondeterminism, and their referential transparency, among other characteristics, make logic progı
interesting candidates for obtaining speedups through parallel execution. At the same time, the fa
the typical applications of logic programming frequently involve irregular computatio ...

Keywords: Automatic parallelization, constraint programming, logic programming, parallelism, pr

9 Automated discovery of scoped memory regions for real-time Java
Morgan Deters, Ron K. Cytron
June 2002 **ACM SIGPLAN Notices , Proceedings of the third international symposium on Me management**, Volume 38 Issue 2 supplement
Full text available: pdf(227.49 KB) Additional Information: full citation, abstract, references, citings, index terms

Advances in operating systems and languages have brought the ideal of reasonably-bounded exec
time closer to developers who need such assurances for real-time and embedded systems applicat
Recently, extensions to the Java libraries and virtual machine have been proposed in an emerging
standard, which provides for specification of release times, execution costs, and deadlines for a re
class of threads. To use such features, the code executing in the thread must never reference s ...

Keywords: garbage collection, memory management, real-time Java, regions, trace-based analy

10 Object combining: A new aggressive optimization for object intensive programs
Ronald Veldema, J. H. Ceriel, F. H. Rutger, E. Henri
November 2002 **Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande**
Full text available: pdf(99.27 KB) Additional Information: full citation, abstract, references, index terms

Object combining tries to put objects together that have roughly the same life times in order to re
strain on the memory manager and to reduce the number of pointer indirections during a prograr
execution. Object combining works by appending the fields of one object to another, allowing alloc
and freeing of multiple objects with a single heap (de)allocation. Unlike object *inlining*, which will
optimize objects where one has a (unique) pointer to another, our optimization al ...

Keywords: Java, garbage collection, object management

11 Space-efficient closure representations
Zhong Shao, Andrew W. Appel
July 1994 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1994 ACM conference on LISP functional programming**, Volume VII Issue 3
Full text available: pdf(1.26 MB) Additional Information: full citation, abstract, references, citings, index terms

Many modern compilers implement function calls (or returns) in two steps: first, a closure environ
properly installed to provide access for free variables in the target program fragment; second, the
is transferred to the target by a "jump with arguments (or results)". Closure conversion, which de
where and how to represent closures at runtime, is a crucial step in compilation of functional langı
We have a new algorithm t ...

12 Fast procedure calls
Butler W. Lampson
March 1982 **Proceedings of the first international symposium on Architectural support for programming languages and operating systems**, Volume 10 , 17 Issue 2 , 4
Full text available: pdf(973.90 KB) Additional Information: full citation, abstract, references, citings, index terms

A mechanism for control transfers should handle a variety of applications (e.g., procedure calls an returns, coroutine transfers, exceptions, process switches) in a uniform way. It should also allow a implementation in which the common cases of procedure call and return are extremely fast, prefe fast as unconditional jumps in the normal case. This paper describes such a mechanism and meth its efficient implementation.

Keywords: Architecture, Call, Frame, Procedure, Registers, Stack, Transfer

13 Implementing jalapeño in Java
Bowen Alpern, C. R. Attanasio, Anthony Cocchi, Derek Lieber, Stephen Smith, Ton Ngo, John J. Bart Susan Flynn Hummel, Janice C. Sheperd, Mark Mergen
October 1999 **ACM SIGPLAN Notices , Proceedings of the 14th ACM SIGPLAN conference on oriented programming, systems, languages, and applications**, Volume 34 Issue 10
Full text available: pdf(1.57 MB) Additional Information: full citation, abstract, references, citings, index terms

Jalapeño is a virtual machine for Java™ servers written in Java.A running Java program involves f layers of functionality: the user code, the virtual-machine, the operating system, and the hardwar drawing the Java / non-Java boundary below the virtual machine rather than above it, Jalapeño re the boundary-crossing overhead and opens up more opportunities for optimization.To get Jalapeñ started, a boot image of a ...

14 A structural view of the Cedar programming environment
Daniel C. Swinehart, Polle T. Zellweger, Richard J. Beach, Robert B. Hagmann
August 1986 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 8 I
Full text available: pdf(6.32 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents an overview of the Cedar programming environment, focusing on its overall structure—that is, the major components of Cedar and the way they are organized. Cedar support development of programs written in a single programming language, also called Cedar. Its primar purpose is to increase the productivity of programmers whose activities include experimental programming and the development of prototype software systems for a high-performance person computer. T ...

15 Creating and preserving locality of java applications at allocation and garbage collection time
Yefim Shuf, Manish Gupta, Hubertus Franke, Andrew Appel, Jaswinder Pal Singh
November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference or Object-oriented programming, systems, languages, and applications**, Volume 3 11
Full text available: pdf(180.20 KB) Additional Information: full citation, abstract, references, citings, index terms

The growing gap between processor and memory speeds is motivating the need for optimization strategies that improve data locality. A major challenge is to devise techniques suitable for pointe intensive applications. This paper presents two techniques aimed at improving the memory behavi pointer-intensive applications with dynamic memory allocation, such as those written in Java. First present an allocation time object placement technique based on the recently introduced notion of

Keywords: *JVM, Java, garbage collection, heap traversal, locality, locality based graph traversal, memory allocation, memory management, object co-allocation, object placement, prolific types, r systems*

16 Escape analysis for Java™: Theory and practice
Bruno Blanchet
November 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 6
Full text available: pdf(684.21 KB) Additional Information: full citation, abstract, references, index terms

Escape analysis is a static analysis that determines whether the lifetime of data may exceed its st scope.This paper first presents the design and correctness proof of an escape analysis for Java™. analysis is interprocedural, context sensitive, and as flow-sensitive as the static single assignment So, assignments to object fields are analyzed in a flow-insensitive manner. Since Java is an imper language, the effect of assignments must be precisely determined. Thi ...

Keywords: Java, optimization, stack allocation, static analysis, synchronization elimination

17 Automatic pool allocation for disjoint data structures
Chris Lattner, Vikram Adve
June 2002 **ACM SIGPLAN Notices , Proceedings of the workshop on Memory system performance**, Volume 38 Issue 2 supplement
Full text available: pdf(1.48 MB) Additional Information: full citation, abstract, references, citings

This paper presents an analysis technique and a novel program transformation that can enable po optimizations for entire linked data structures. The fully automatic transformation converts ordina programs to use pool (aka region) allocation for heap-based data structures. The transformation r an efficient link-time interprocedural analysis to identify disjoint data structures in the program, tc whether these data structures are accessed in a type-safe manner, and to constru ...

18 A high performance Erlang system
Erik Johansson, Mikael Pettersson, Konstantinos Sagonas
September 2000 **Proceedings of the 2nd ACM SIGPLAN international conference on Principles practice of declarative programming**
Full text available: pdf(320.62 KB) Additional Information: full citation, references, citings, index terms

19 Practicing JUDO: Java under dynamic optimizations
Michał Cierniak, Guei-Yuan Lueh, James M. Stichnoth
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation**, Volume 35 Issue 5
Full text available: pdf(190.06 KB) Additional Information: full citation, abstract, references, citings, index terms

A high-performance implementation of a Java Virtual Machine (JVM) consists of efficient implemen of Just-In-Time (JIT) compilation, exception handling, synchronization mechanism, and garbage collection (GC). These components are tightly coupled to achieve high performance. In this paper, present some static anddynamic techniques implemented in the JIT compilation and exception har of the Microprocessor Research Lab Virtual Machine (MRL VM), ...

20 Distributed systems - programming and management: On remote procedure call
Patrícia Gomes Soares
November 1992 **Proceedings of the 1992 conference of the Centre for Advanced Studies on Collaborative research - Volume 2**
Full text available: pdf(4.52 MB) Additional Information: full citation, abstract, references

The Remote Procedure Call (RPC) paradigm is reviewed. The concept is described, along with the backbone structure of the mechanisms that support it. An overview of works in supporting these

mechanisms is discussed. Extensions to the paradigm that have been proposed to enlarge its suita
are studied. The main contributions of this paper are a standard view and classification of RPC
mechanisms according to different perspectives, and a snapshot of the paradigm in use today and
goals for t ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next